Procesamiento de Lenguaje Natural Avanzado

Representaciones contextualizadas del lenguaje,

Dra. Helena Gómez Adorno
helena.gomez@iimas.unam.mx

Dr. Fazlourrahman Balouchzahi
fbalouc@iimas.unam.mx

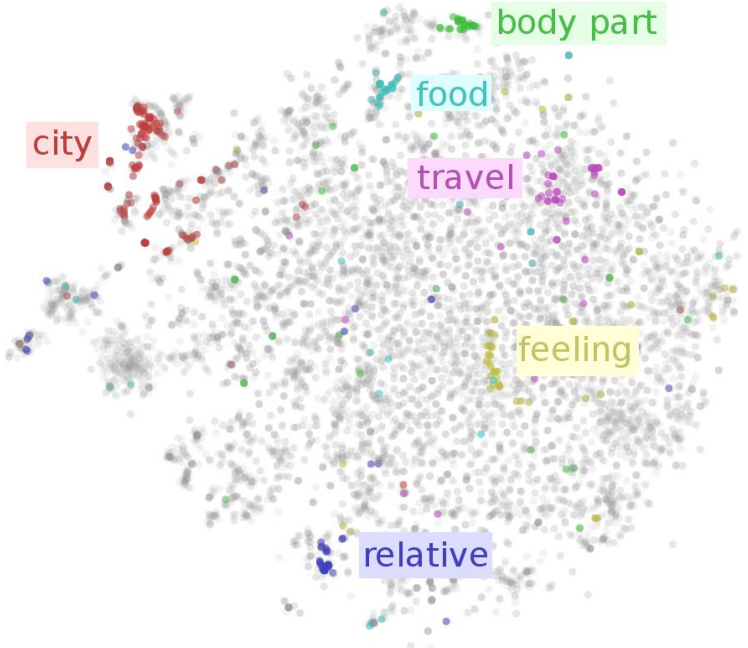
Correo del curso:
pln.cienciadedatos@gmail.com

Vectores de Palabras

Las técnicas clásicas de representación de palabras consideran el uso de vectores dispersos de gran dimensión.

Algunas representaciones como el one-hot hacen complicado establecer una similitud entre palabras.

Utilizar vectores con menor dimensión ayuda a mejorar el rendimiento de los algoritmos de aprendizaje automatizado, con la desventaja de perder interpretabilidad.



Hipótesis distribucional

J.R.Firth 1957

- "You shall know a word by the company it keeps"
- One of the most successful ideas of modern statistical NLP!

Algunos métodos para generar vectores de palabras se basan en la idea de que una palabra puede ser caracterizada por las palabras que la acompañan.

Las palabras que aparecen o son usadas en contextos similares tienen características semánticas similares.

Hipótesis distribucional

“tejuino”

C1: Una botella de ___ está en la mesa.

C2: A todo el mundo le gusta ___.

C3: No debes tomar ___ antes de conducir.

C4: Podemos hacer ___ a partir del maiz.

Hipótesis distribucional

C1: Una botella de ___ está en la mesa.

C2: A todo el mundo le gusta el ___.

C3: No debes tomar ___ antes de conducir.

C4: Podemos hacer ___ a partir del maiz.

	C1	C2	C3	C4
tejuino	1	1	1	1
fuerte	0	0	0	0
aceite de motor	1	0	0	0
tortillas	0	0	0	1
vino	1	1	1	0
pan	0	1	0	0

Palabras de ocurren en contextos similares tienen a tener significado similar

Palabras como vectores

Construiremos un nuevo modelo de significado centrado en la similitud.

- Cada palabra es un vector.
- Las palabras similares están "cercanas en el espacio".
- Una primera solución: podemos usar vectores de contexto para representar el significado de las palabras.
- Matriz de coocurrencia palabra-palabra:

	aardvark	computer	data	pinch	result	sugar	…
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	

Palabras como vectores

Problema: Usar conteos de frecuencia sin procesar no siempre es muy recomendable.

- Solución: ¡Ponderemos los conteos!
- PPMI = Información Mutua Puntual Positiva

$$\text{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0)$$

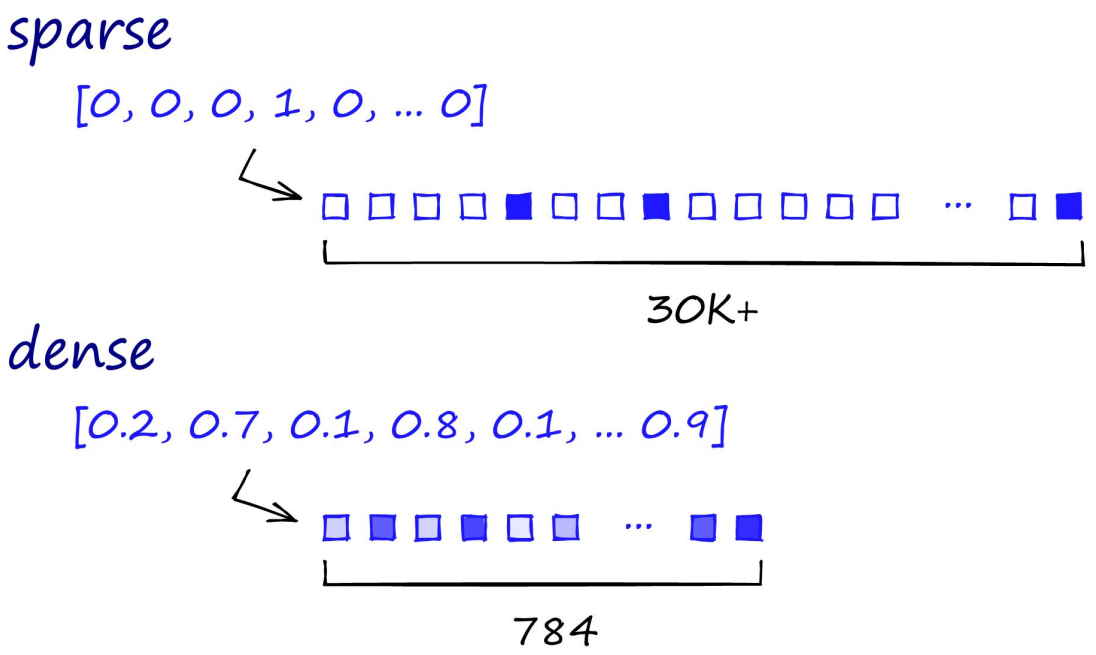
	computer	data	result	pie	sugar
cherry	2	8	9	442	25
strawberry	0	0	1	60	19
digital	1670	1683	85	5	4
information	3325	3982	378	5	13

	computer	data	result	pie	sugar
cherry	0	0	0	4.38	3.30
strawberry	0	0	0	4.10	5.51
digital	0.18	0.01	0	0	0
information	0.02	0.09	0.28	0	0

Vectores densos vs. dispensos (dense vs sparse)

Aun así, los vectores que obtenemos de la matriz de ocurrencia palabra-palabra son dispersos (la mayoría son ceros) y largos (tamaño del vocabulario).

- **Alternativa**: queremos representar las palabras como vectores **cortos** (de 50 a 300 dimensiones) y **densos** (de valor real).
- La base de todos los sistemas modernos de PLN.

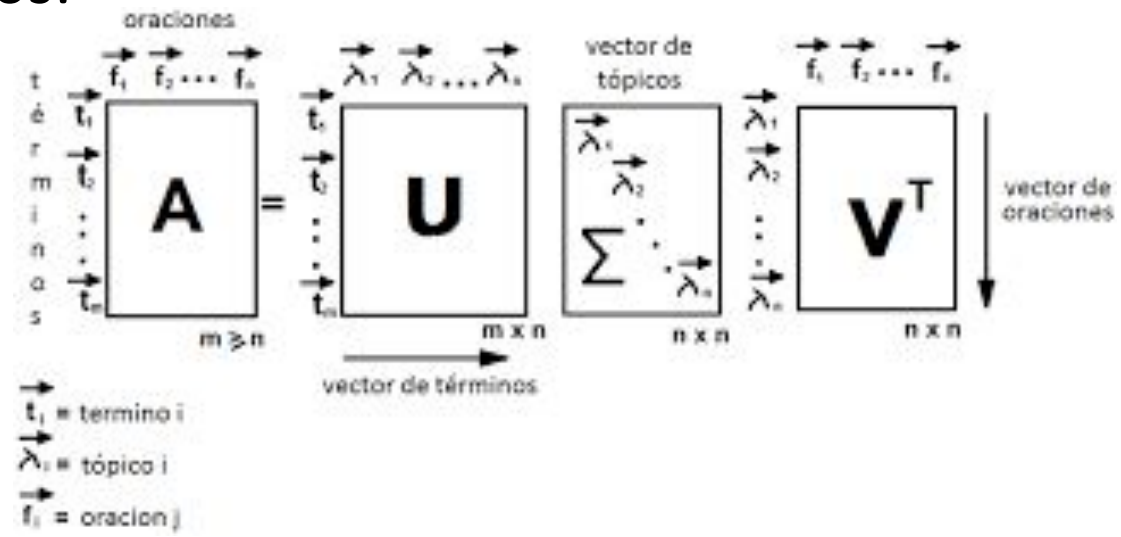


Porque vectores densos?

Los vectores cortos son más fáciles de usar como características en sistemas de aprendizaje automático (ML).

- Los vectores densos pueden generalizarse mejor que almacenar conteos explícitos.
- Son más eficaces para capturar sinonimia.
    - w1 co-ocurre con "car", w2 co-ocurre con "automobile".

- Diferentes métodos para obtener vectores densos:
    - Descomposición en valores singulares (SVD)
    - word2vec y amigos: ¡aprende los vectores!



Word2Vec

Propone un método para la creación de vectores densos de palabras utilizando una red neuronal.

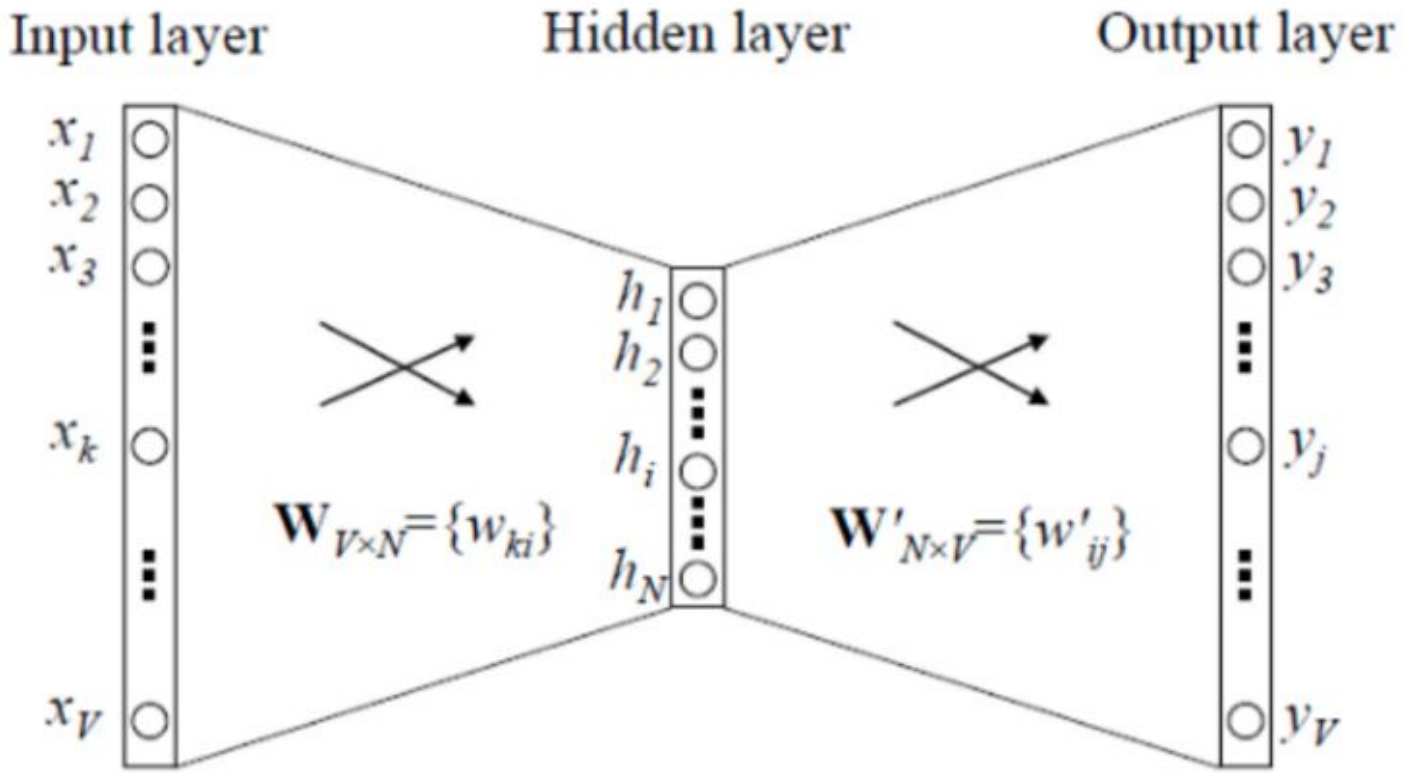
La tarea de entrenamiento consiste en predecir una palabra dada su contexto.

Para el entrenamiento de los vectores se puede utilizar cualquier corpus textual. No es necesario que esté anotado.

- Se utilizan parejas de palabras-contexto como ejemplos positivos.
- Se muestrean ejemplos negativos utilizando palabras que no aparecen en el contexto de la palabra.

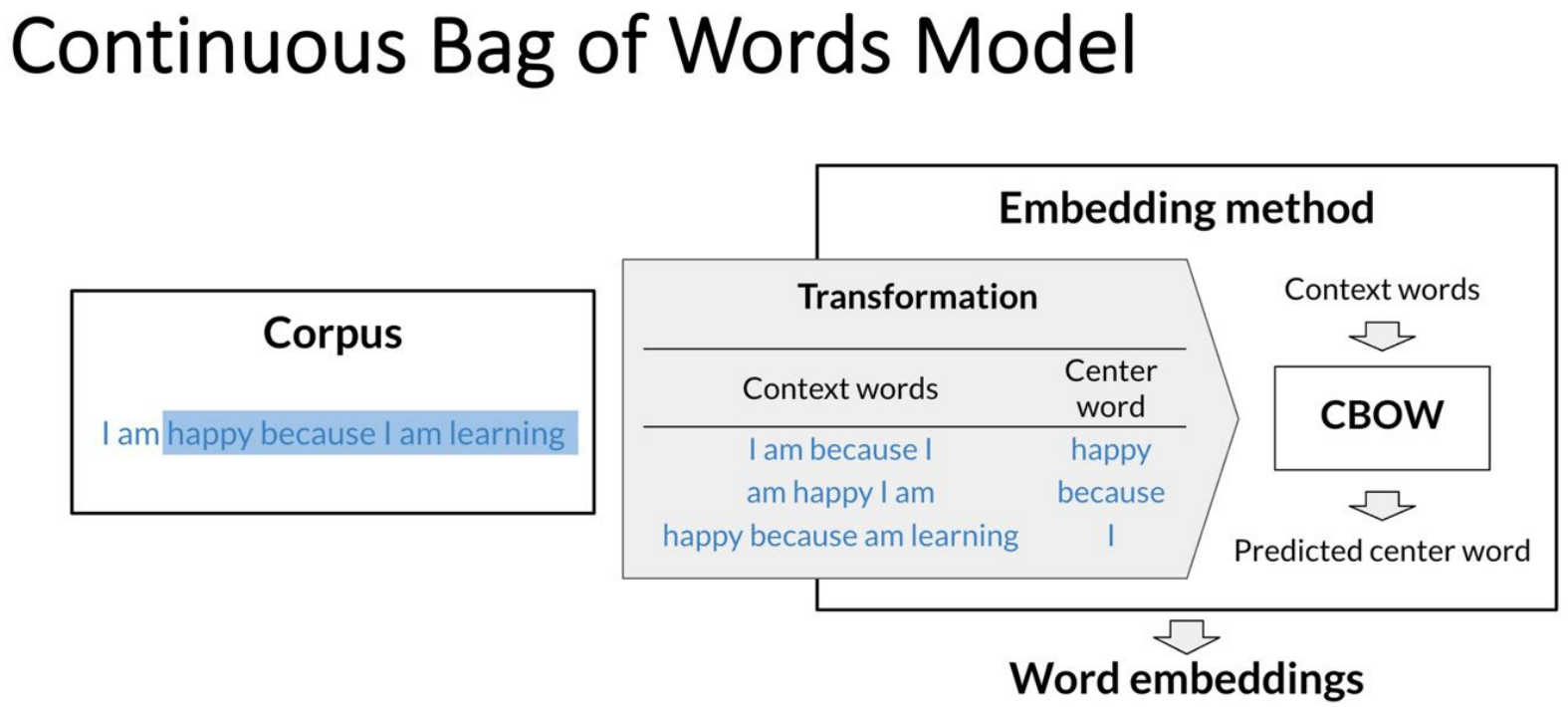
Mikolov 2013 https://arxiv.org/abs/1301.3781

CBOW



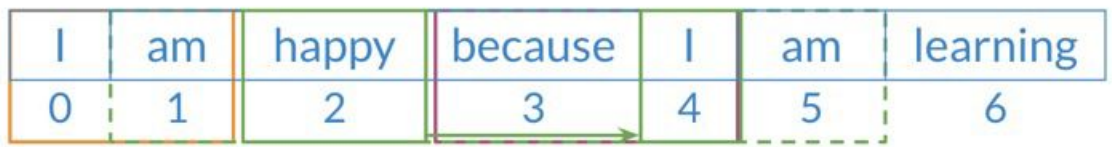
Mikolov 2013 https://arxiv.org/abs/1301.3781

COntinuous Bag of words (CBOW)



Sliding Window of words in Python

```
def get_windows(words, C):
    i = C
    while i < len(words) - C:
        center_word = words[i]
        context_words = words[(i - C):i] + words[(i+1):(i+C+1)]
        yield context_words, center_word
        i += 1
```

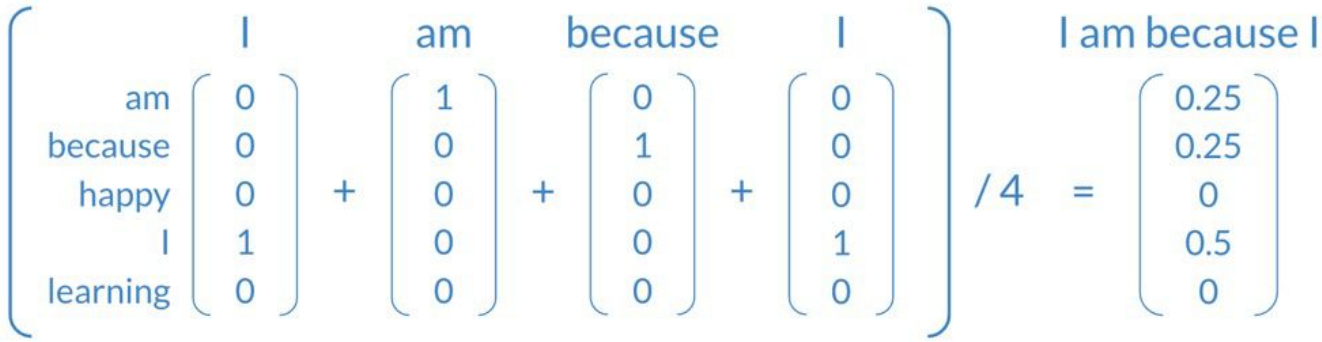


- The code above shows you a function which takes in two parameters.
- Words: a list of words.
- C: the context size.
- We first start by setting *i* to C. Then we single out the center_word, and the context_words. We then yield those and increment *i*.

COntinuous Bag of words (CBOW)

Transforming Words into Vectors

- To transform the context vectors into one single vector, you can use the following.



- As you can see, we started with one-hot vectors for the context words and and we transform them into a single vector by taking an average. As a result you end up having the following vectors that you can use for your training.

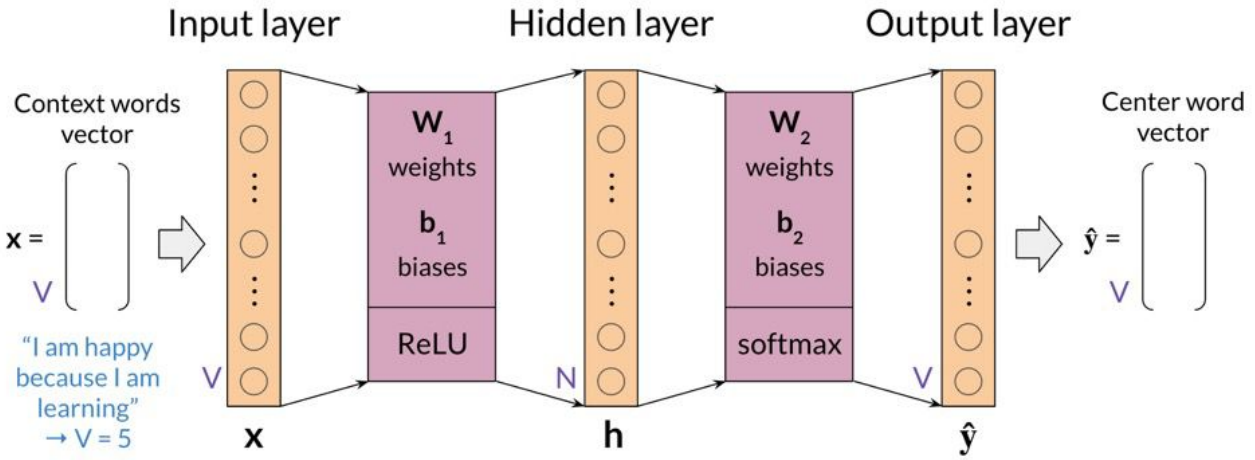
Context words	Context words vector	*Center word*	Center word vector
I am because I	[0.25; 0.25; 0; 0.5; 0]	*happy*	[0; 0; 1; 0; 0]

14

COntinuous Bag of words (CBOW)

Architecture for the CBOW Model

- The architecture for the CBOW model could be described as follows



- You have an input, *X,* which is the average of all context vectors. You then multiply it by W1 and add b1. The result goes through a ReLU function to give you your hidden layer. That layer is then multiplied by W2 and you add b2. The result goes through a softmax which gives you a distribution over *V,* vocabulary words. You pick the vocabulary word that corresponds to the arg-max of the output.

COntinuous Bag of words (CBOW)

Architecture of the CBOW Model: Dimensions

- The equations for the previous model are:

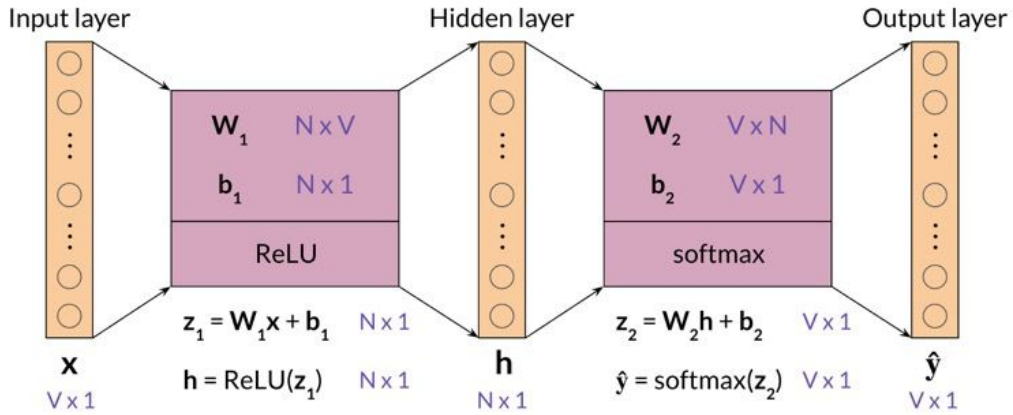
$$z_1 = W_1 x + b_1$$

$$h = ReLU(z_1)$$

$$z_2 = W_2 h + b_2$$

$$\hat{y} = softmax(z_2)$$

- Here, you can see the dimensions:

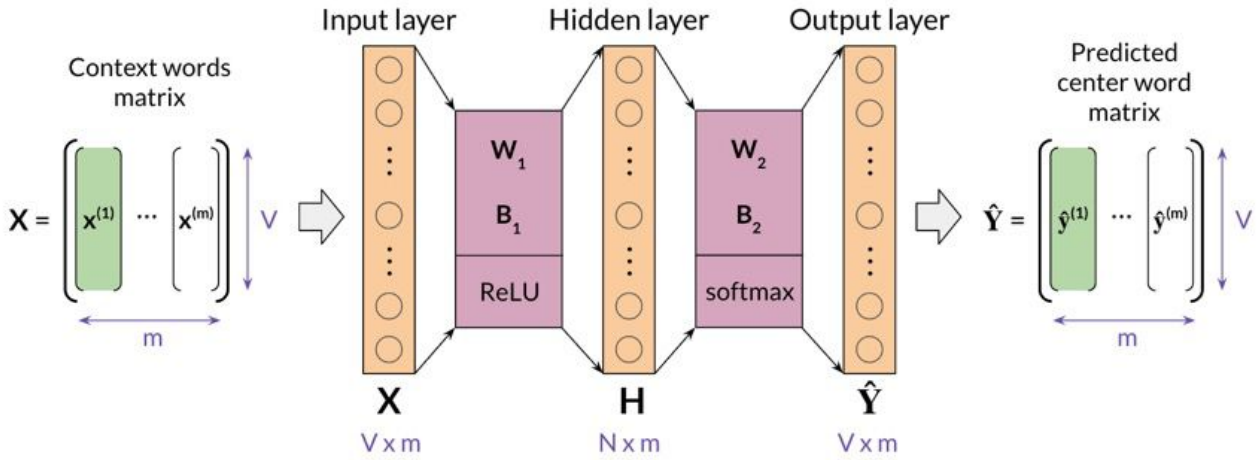


- Make sure you go through the matrix multiplications and understand why the dimensions make sense.

COntinuous Bag of words (CBOW)

Architecture of the CBOW Model: Dimensions

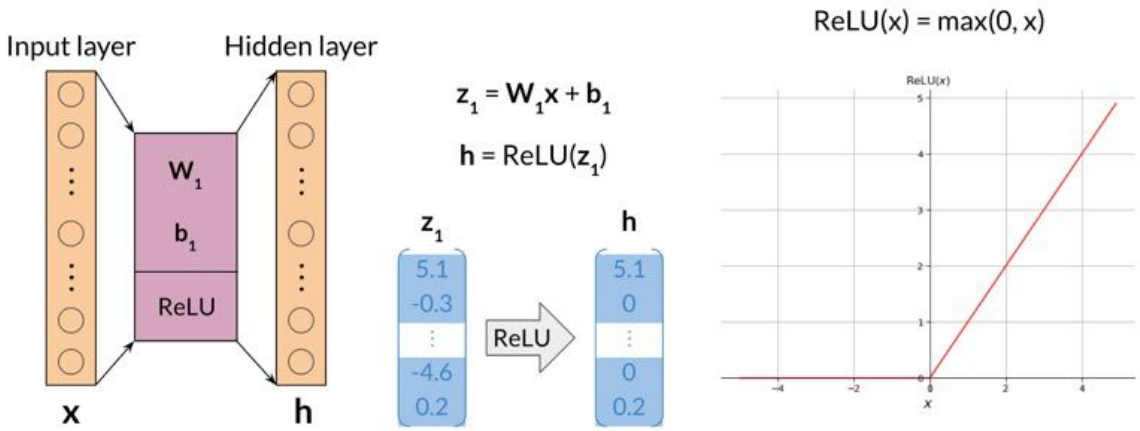
- When dealing with batch input, you can stack the examples as columns. You can then proceed to multiply the matrices as follows:



- In the diagram above, you can see the dimensions of each matrix. Note that your Y^ is of dimension V by m. Each column is the prediction of the column corresponding to the context words. So the first column in Y^ is the prediction corresponding to the first column of *X*.

Architecture of the CBOW Model: Activation Functions

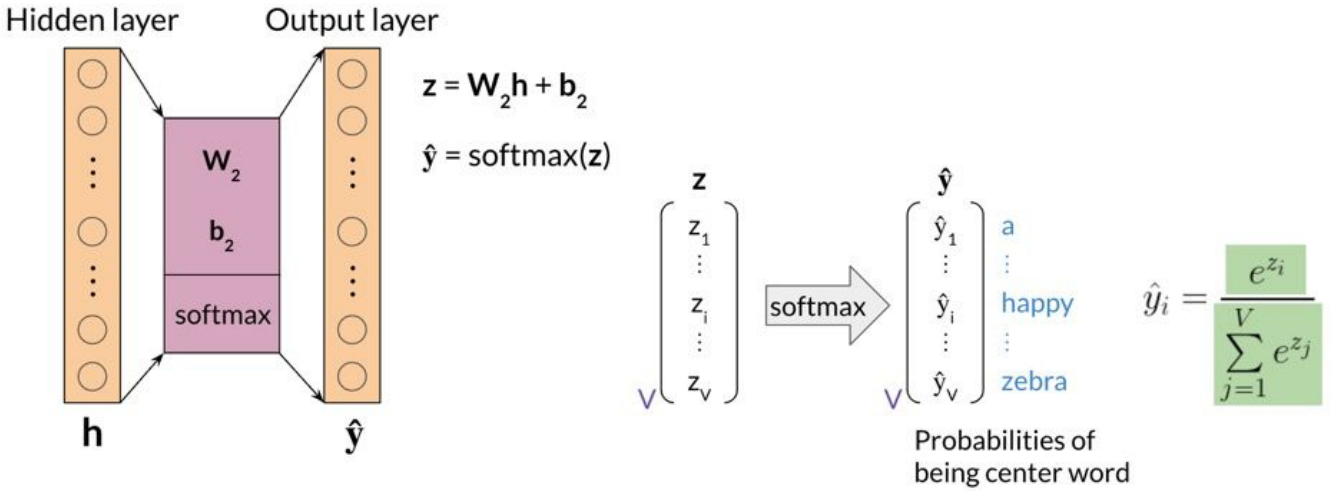
- **ReLU funciton**
- The rectified linear unit (ReLU), is one of the most popular activation functions. When you feed a vector, namely x, into a ReLU function. You end up taking x=max(0,x). This is a drawing that shows ReLU.



COntinuous Bag of words (CBOW)

Architecture of the CBOW Model: Activation Functions

- **Softmax function**
- The softmax function takes a vector and transforms it into a probability distribution. For example, given the following vector z, you can transform it into a probability distribution as follows.



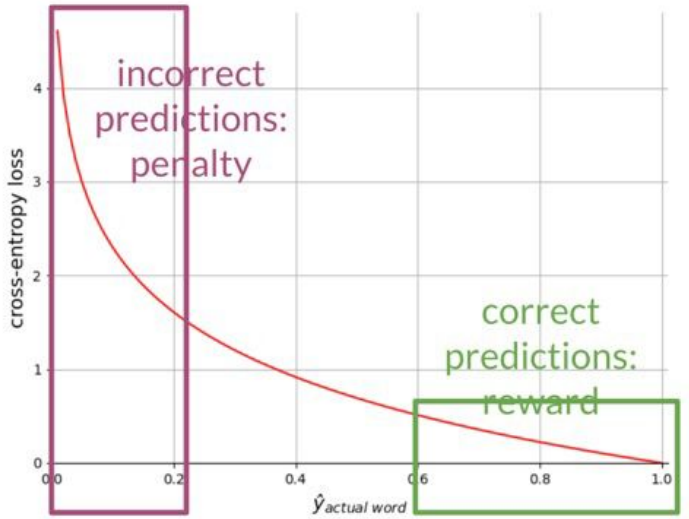
Training a CBOW Model: Cost Function

- The cost function for the CBOW model is a cross-entropy loss defined as: $J = -\sum_{k=1}^{V} y_k \log \hat{y}_k$
- Here is an example where you use the equation above.

$$J = -\log \hat{y}_{\text{actual word}}$$

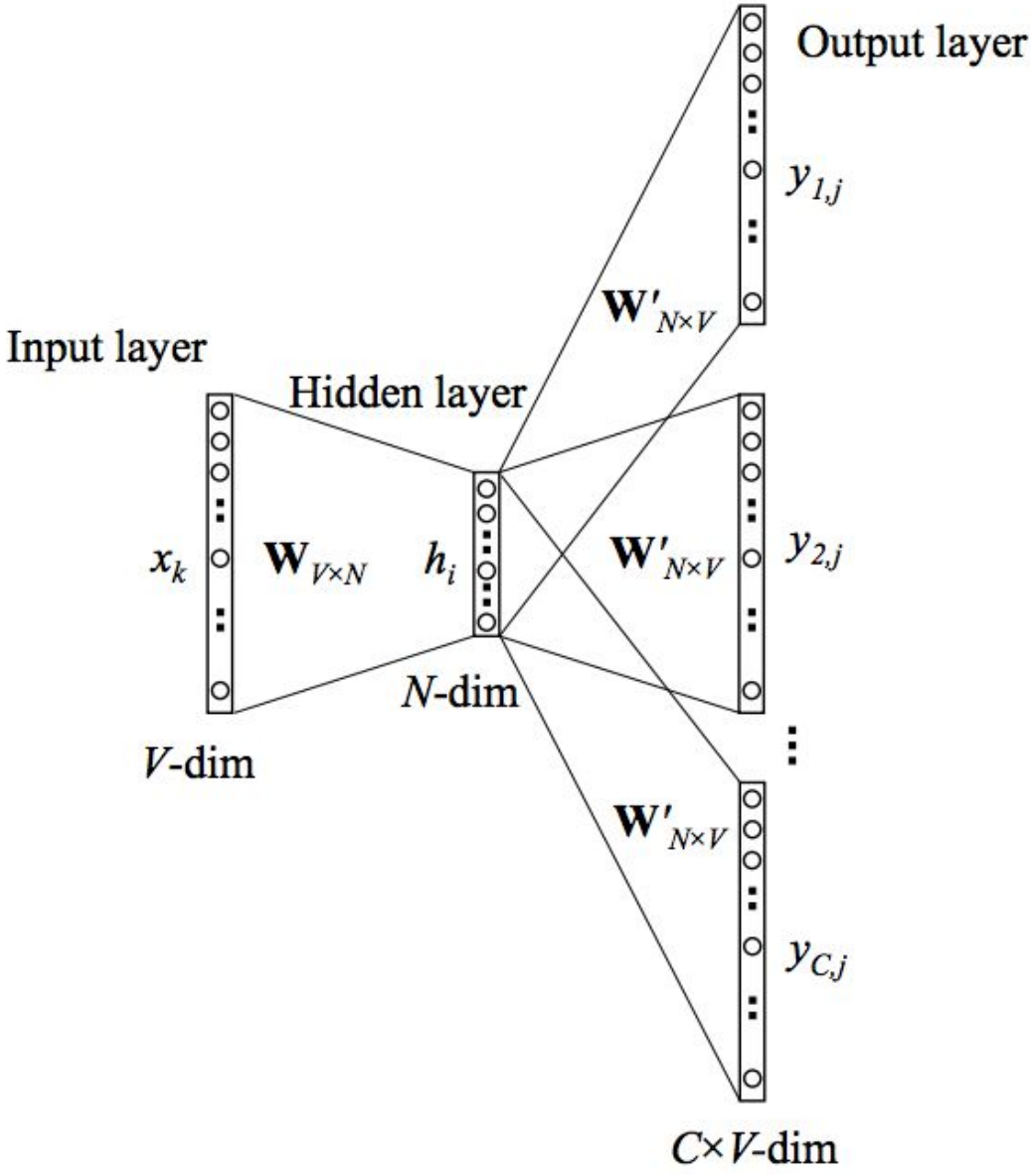
y		$\hat{y}$
0	am	0.96
0	because	0.01
1	happy	0.01
0	I	0.01
0	learning	0.01

→ J = 4.61



- Why is the cost 4.61 in the example above?

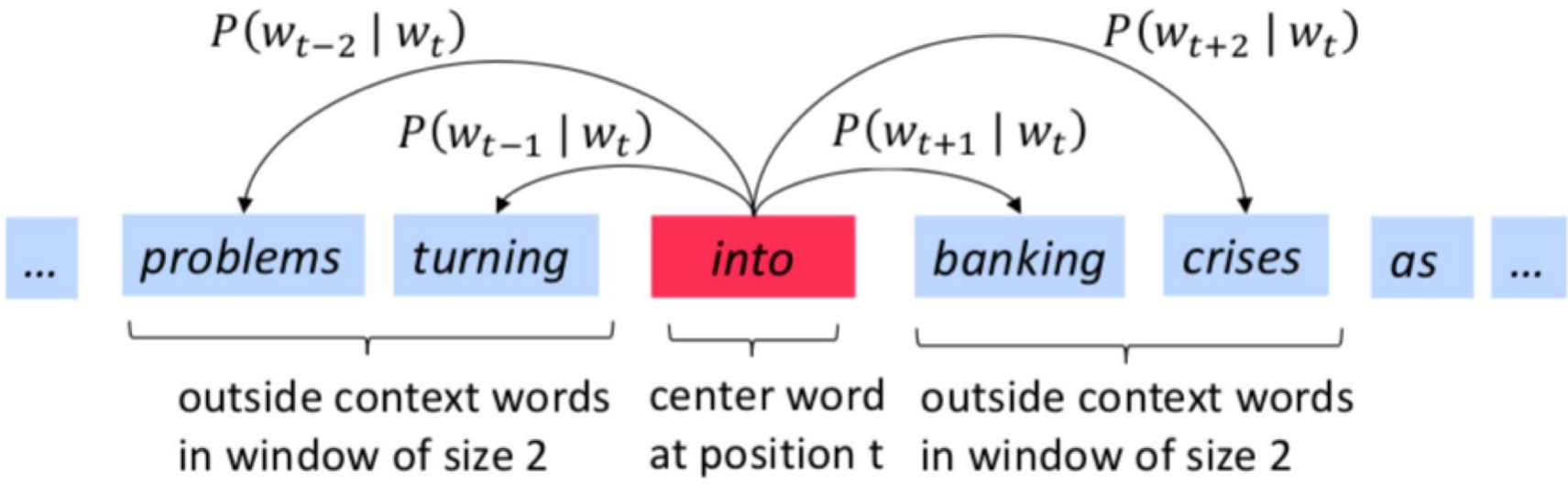
Skip-gram


Output layer
$y_{1,j}$
$\mathbf{W}'_{N\times V}$
Input layer
Hidden layer
x_k
$\mathbf{W}_{V\times N}$
h_i
$\mathbf{W}'_{N\times V}$
$y_{2,j}$
N-dim
V-dim
$\mathbf{W}'_{N\times V}$
$y_{C,j}$
$C\times V$-dim

Skip-gram

La idea: queremos usar palabras para predecir su contexto.

- **Contexto**: una ventana fija de *2m*.



Skip-gram

Para cada posición *t* = 1, 2,…T, predecir palabras de contexto dentro del tamaño de contexto ***m***, dada la palabra central w_j y todos los parámetros a optimizar.

all the parameters to be optimized

$$\mathcal{L}(\theta) = \prod_{t=1}^{T} \prod_{-m \le j \le m, j \neq 0} P(w_{t+j} \mid w_t; \theta)$$

La función objetivo J(θ) es el negativo del logaritmo de verosimilitud (promedio):

$$J(\theta) = -\frac{1}{T} \log \mathcal{L}(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-m \le j \le m, j \neq 0} \log P(w_{t+j} \mid w_t; \theta)$$

¿Cómo definir? $P(w_{t+j} \mid w_t; \theta)$

Tenemos dos conjuntos de vectores para cada palabra del vocabulario.

$$\mathbf{u}_i \in \mathbb{R}^d \text{ : embedding for target word } i$$

$$\mathbf{v}_{i'} \in \mathbb{R}^d \text{ : embedding for context word } i'$$

Usamos el producto interno $\mathbf{u}_i \cdot \mathbf{v}_{i'}$ para medir la probabilidad de que la palabra *i* aparezca con la palabra de contexto *i'*. Cuanto más grande, mejor.

"softmax" we learned last time!

$$P(w_{t+j} \mid w_t) = \frac{\exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_{w_{t+j}})}{\sum_{k \in V} \exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_k)}$$

$\theta = \{\{\mathbf{u}_k\}, \{\mathbf{v}_k\}\}$ are all the parameters in this model!

¿Cómo entrenar el modelo?

Calculando todos los gradientes juntos!

$$\theta = \{\{\mathbf{u}_k\}, \{\mathbf{v}_k\}\}$$

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0} \log P(w_{t+j} \mid w_t; \theta) \qquad \nabla_\theta J(\theta) = ?$$

Podemos aplicar el Descenso del Gradiente Estocástico

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta J(\theta)$$

Ensamblando todo

- Entrada: corpus de texto, tamaño de contexto m, tamaño de incrustación d, V
- Inicializar ! aleatoriamente u_i, v_i
- Recorrer el corpus de entrenamiento y recopilar datos de entrenamiento (t, c):

  - Update $\mathbf{u}_t \leftarrow \mathbf{u}_t - \eta \frac{\partial y}{\partial \mathbf{u_t}}$
  - Update $\mathbf{v}_k \leftarrow \mathbf{v}_k - \eta \frac{\partial y}{\partial \mathbf{v_k}}, \forall k \in V$

Skip-gram with negative sampling (SGNS)

- **Problema**: cada vez que se obtiene un par de (t, c), se debe usar v_k con todas las palabras del vocabulario! Es muy costoso computacionalmente.

$$\frac{\partial y}{\partial \mathbf{u}_t} = -\mathbf{v}_c + \sum_{k \in V} P(k \mid t)\mathbf{v}_k$$

$$\frac{\partial y}{\partial \mathbf{v}_k} = -1(k = c)\mathbf{u}_t + P(k \mid t)\mathbf{u}_t$$

Muestreo negativo: en lugar de considerar todas las palabras en V, muestreamos aleatoriamente K (5-20) ejemplos negativos.

$$\textbf{softmax:} \qquad y = -\log\left(\frac{\exp(\mathbf{u}_t \cdot \mathbf{v}_c)}{\sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k)}\right)$$

$$\textbf{NS:} \qquad y = -\log(\sigma(\mathbf{u}_t \cdot \mathbf{v}_c)) - \sum_{i=1}^{K} \mathbb{E}_{j \sim P(w)} \log(\sigma(-\mathbf{u}_t \cdot \mathbf{v}_j))$$

Word2Vec

- El tamaño de la ventana es un hiperparámetro que es necesario ajustar.
- Se recomienda un tamaño entre 2 y 20 palabras.
- Si se utiliza un tamaño pequeño se favorece que las palabras sean más cercanas a palabras que aparecen en contextos similares.
- Si se utiliza un tamaño más grande se favorece que las palabras más cercanas sean aquellas que son del mismo tema, pero que no pertenecen a la misma categoría.

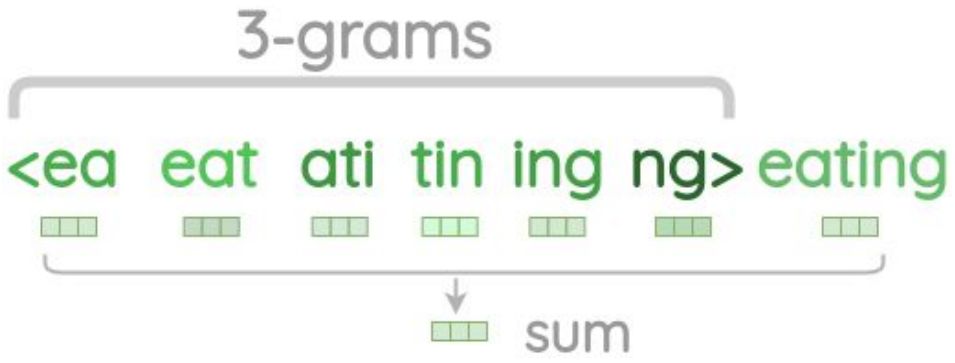
Word2Vec

```
from gensim.models import Word2Vec

corpus = [
      'Text of the first document.’,
      'Text of the second document made longer.’,
      'Number three.’,
      'This is number four.',
]

tokenized_sentences = [sentence.split() for sentence in corpus]
model = Word2Vec(tokenized_sentences, min_count=1)
```

https://radimrehurek.com/gensim/models/word2vec.html

FastText

- Es una extensión al modelo skip-gram.
- Utiliza un modelo de subpalabras, cada palabra se representa como un conjunto de subpalabras o n-gramas de caracteres.
- El algoritmo skip-gram se ejecuta sobre estas subpalabras, al final el vector de la palabra se obtiene sumando los vectores de las subpalabras.
- Es más robusto a palabras fuera del vocabulario.



Bojanowski 2017 https://arxiv.org/pdf/1607.04606.pdf

GloVe

- Este método considera al igual que skip-gram una ventana de contexto.
- Toma en cuenta estadísticas generales del corpus, utiliza la matriz de co-ocurrencia.

$$X_{ij} = \frac{P(w_i|w_j)}{P(w_j)}$$

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + b_j - \log X_{ij})^2$$

Pennington 2014 https://nlp.stanford.edu/pubs/glove.pdf

Modelos de embedding pre-entrenados

- word2vec: https://code.google.com/archive/p/word2vec/
- GloVe: https://nlp.stanford.edu/projects/glove/
- FastText: https://fasttext.cc/

Evaluación Extrinseca vs Intrinseca

Evaluación extrínseca

• Conectemos estas incrustaciones de palabras a un sistema de PLN real y veamos si esto mejora el rendimiento.

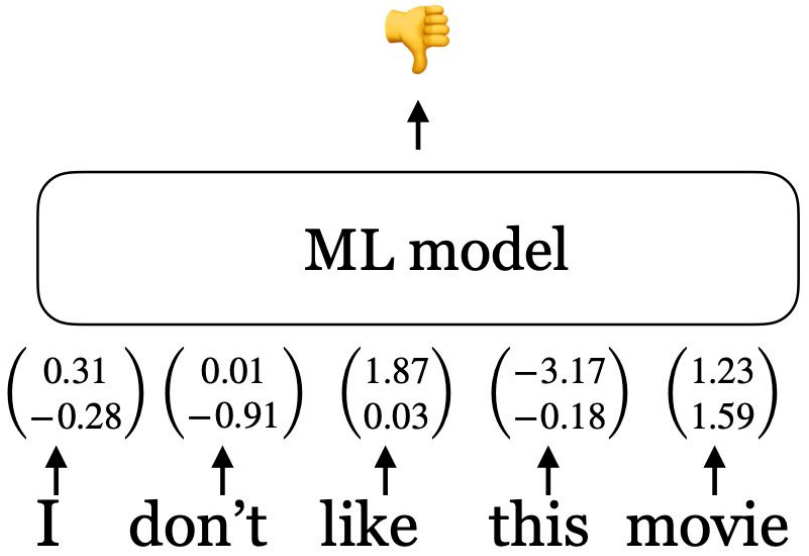
• Puede llevar mucho tiempo, pero sigue siendo la métrica de evaluación más importante.

Evaluación intrínseca

• Evaluar en una subtarea específica/intermedia

• Rápido de calcular

• No está claro si realmente ayuda en la tarea posterior



Evaluación Intrínseca

Similitud de palabras

Conjunto de datos de ejemplo: wordsim-353

353 pares de palabras con juicio humano

https://gabrilovich.com/resources/data/wordsim353/wordsim353.html

Word 1	Word 2	Human (mean)
tiger	cat	7.35
tiger	tiger	10
book	paper	7.46
computer	internet	7.58
plane	car	5.77
professor	doctor	6.62
stock	phone	1.62
stock	CD	1.31
stock	jaguar	0.92

Similitud coseno:

$$\cos(\boldsymbol{u}_i, \boldsymbol{u}_j) = \frac{\boldsymbol{u}_i \cdot \boldsymbol{u}_j}{||\boldsymbol{u}_i||_2 \times ||\boldsymbol{u}_j||_2}.$$

Metrica: correlación

Spearman rank

Vectores basados en Contexto

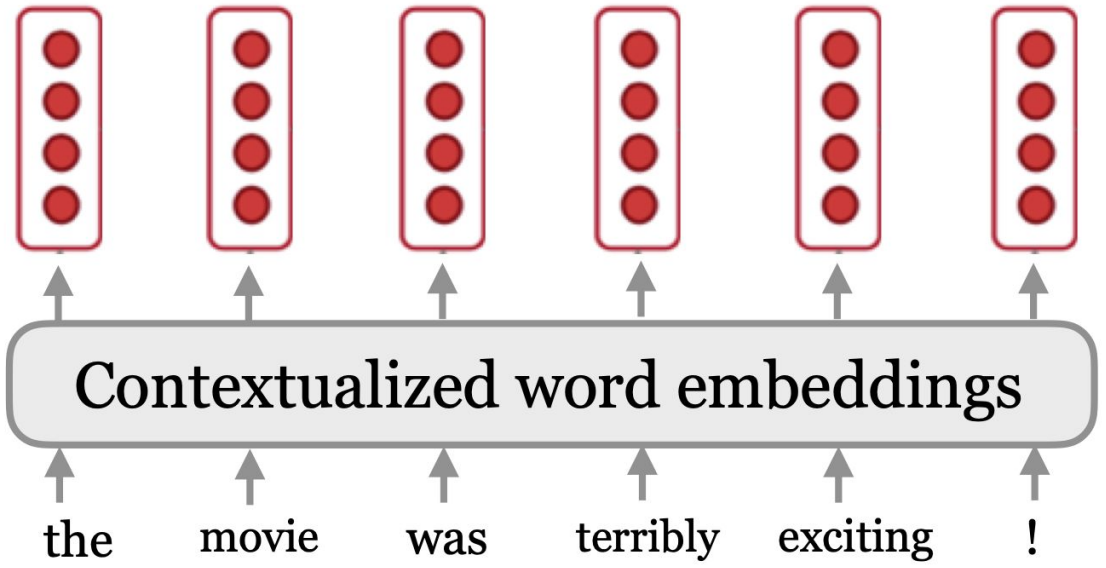
- Los modelos presentados hasta ahora generan vectores estáticos.
- Estas representaciones no toman en cuenta las posibles diferencias en el significado de una misma palabra.
- El contexto puede hacer que las palabras varíen su significado ligeramente o radicalmente.
- Un modelo basado en contexto genera un vector distinto dependiendo del contexto en el que aparece la palabra.

Roget Category	Target Word in Context
FISH/INSECT	. . . fish as Pacific salmon and striped **bass** and. . .
FISH/INSECT	. . . produce filets of smoked **bass** or sturgeon. . .
MUSIC	. . . exciting jazz **bass** player since Ray Brown. . .
MUSIC	. . . play **bass** because he doesn't have to solo. . .

Vectores basados en Contexto

- Construye un vector para cada palabra según su contexto!

= la representación de cada token es una función de la oración de entrada completa.



$$g : (w_1, w_2, \ldots, w_n) \longrightarrow \mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$$

Vectores basados en Contexto

- Cálculo de vector contextual

$$\mathbf{c}_k = f(w_k \mid w_1, w_2, \ldots, w_n) \in \mathbb{R}^d$$

$$f(\text{play} \mid \text{Elmo and Cookie Monster play a game})$$

$$\neq$$

$$f(\text{play} \mid \text{The Broadway play premiered yesterday})$$

Vectores basados en Contexto

Noah Smith. 2019.
Contextual Word Representations: A Contextual Introduction

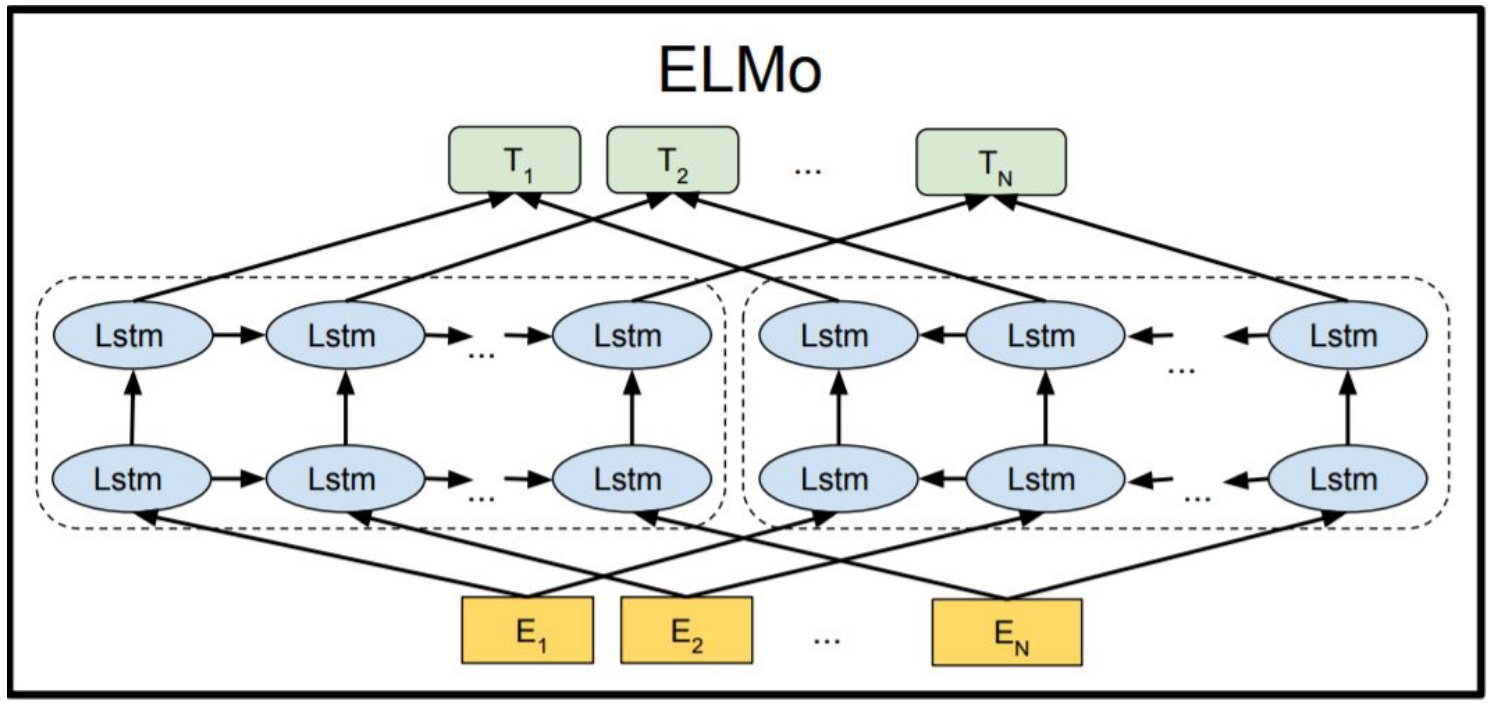
"With hindsight, we can now see that by representing word types independent of context, we were solving a problem that was harder than it needed to be. Because words mean different things in different contexts, we were requiring that type representations capture all of the possibilities. Moving to word token vectors simplifies things, asking the word token representation to capture only what a word means in this context. For the same reasons that the collection of contexts a word type is found in provide clues about its meaning(s), a particular token's context provides clues about its specific meaning."

ELMo

Idea clave:

• Entrenar un modelo del lenguaje (LM) basado en LSTM directo y un LM basado en LSTM inverso en un corpus grande.

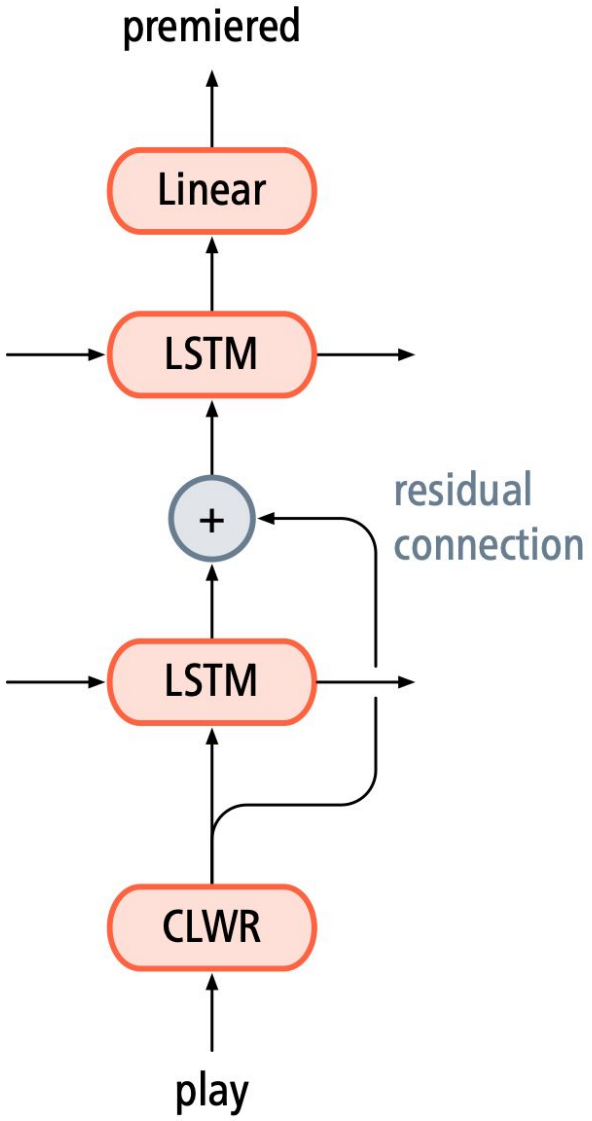
• Usar los estados ocultos de los LSTM de cada token para calcular una representación vectorial de cada palabra.



Peters 2018 https://arxiv.org/abs/1802.05365

Arquitectura ELMo

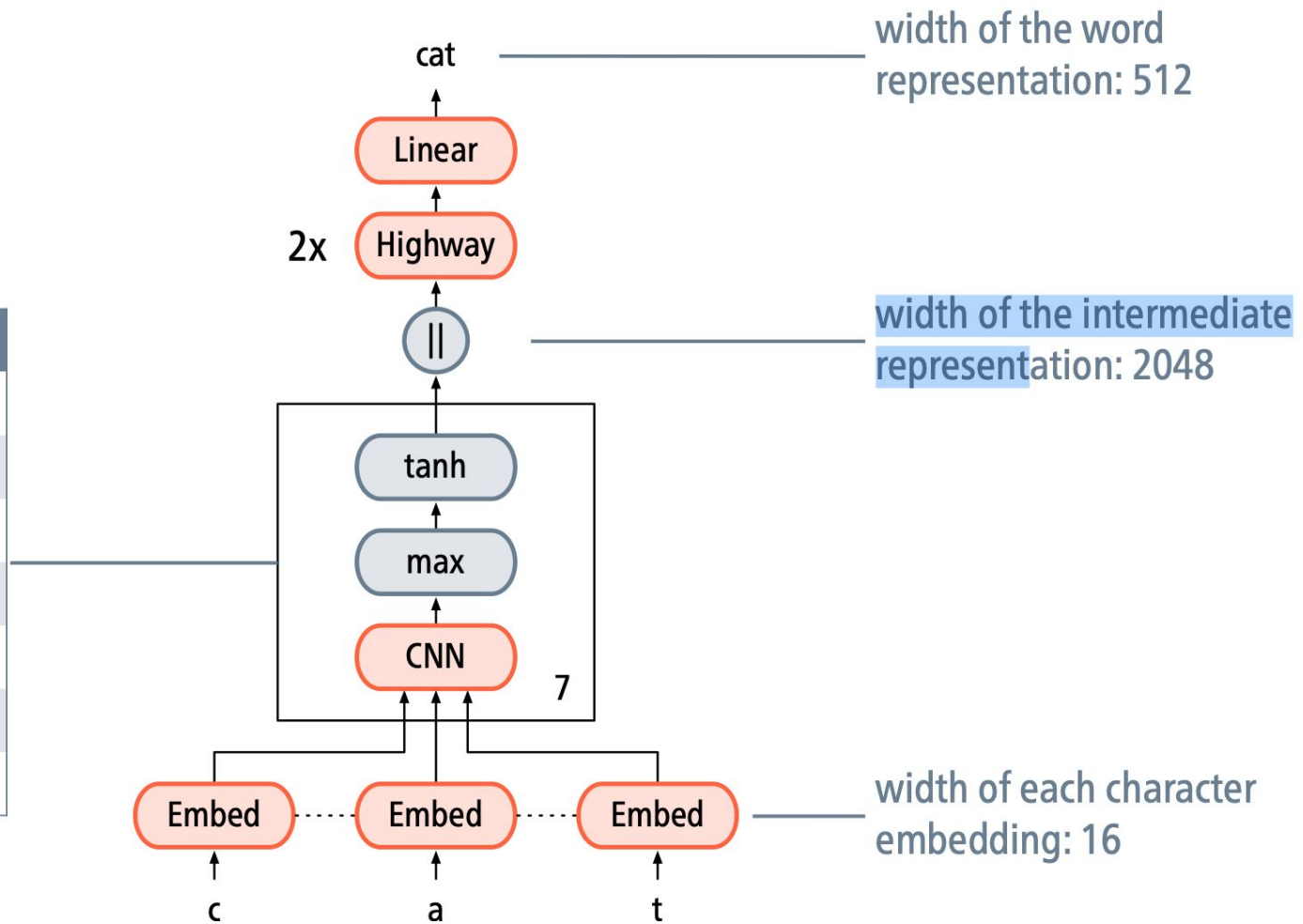
- Dos capas LSTM bidireccionales con una conexión residual entre ellas LSTM
- Representación de palabras insensible al contexto mediante convoluciones de caracteres + LSTM de conexión residual
- La capa somax final calcula una distribución de probabilidad sobre los siguientes tokens



Representaciones de palabras en ELMo

Filter specification

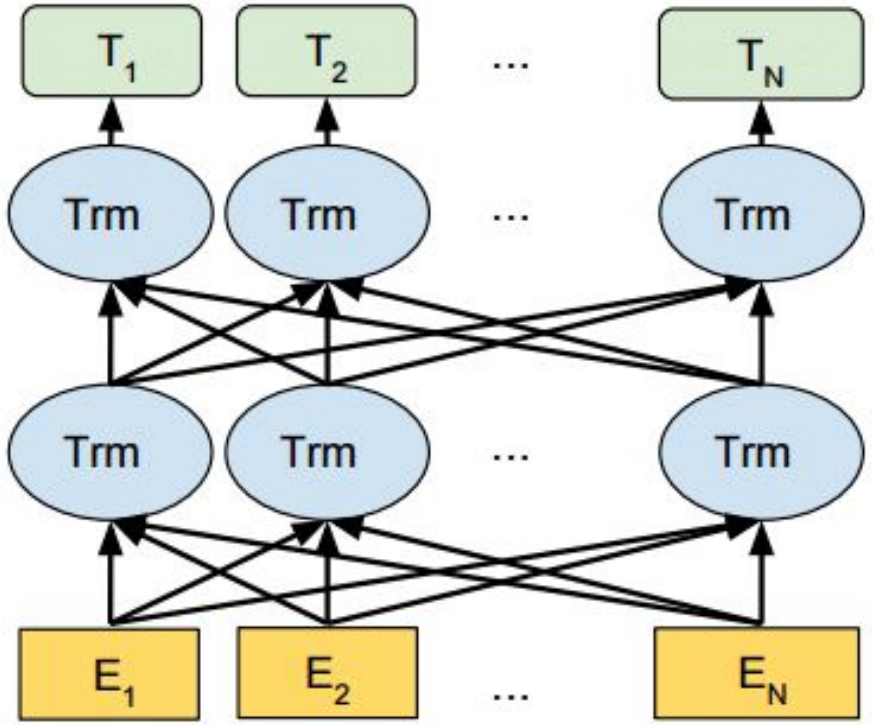
Width	Channels
1	32
2	32
3	64
4	128
5	256
6	512
7	1024



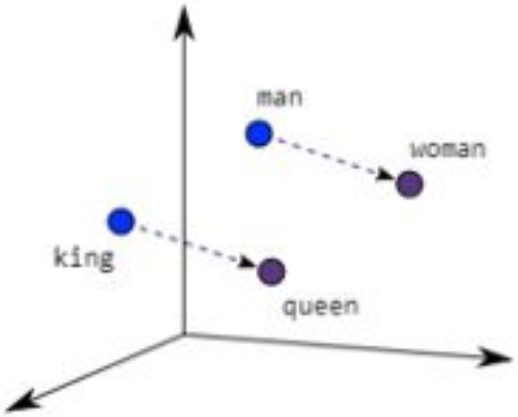
BERT

Modelo de lenguaje preentrenado basado en la arquitectura Transformer.

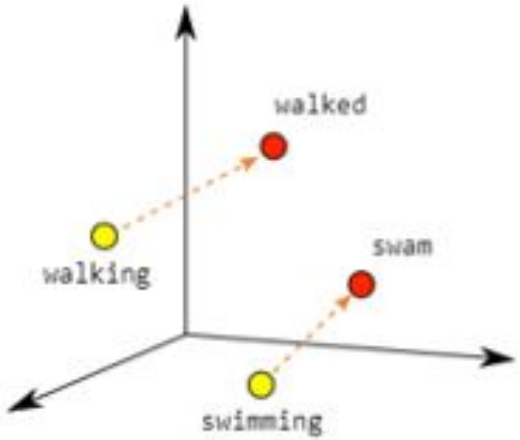
Genera embeddings basados en el contexto.



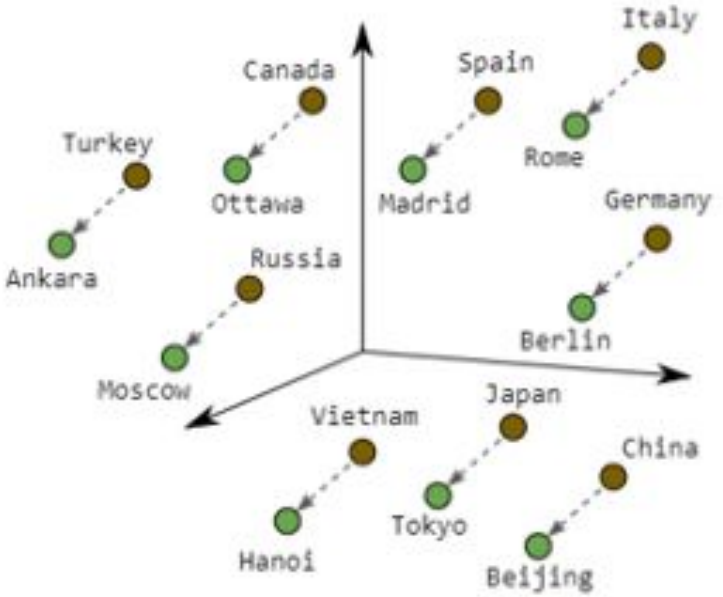
Propiedades de los Word Embeddings



Male-Female

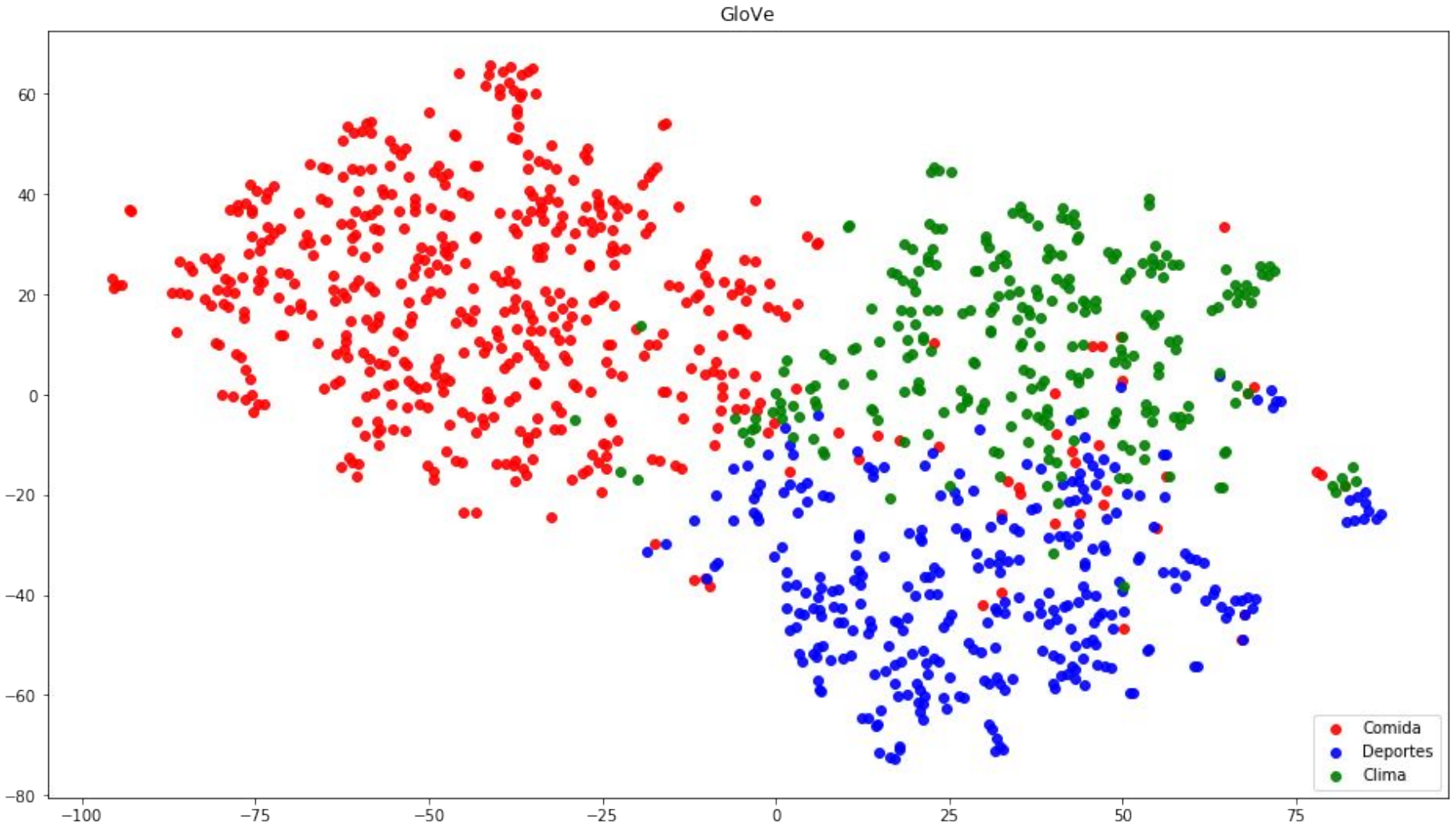


Verb Tense



Country-Capital

Propiedades de los Word Embeddings



Sesgo en Word Embeddings